テーマ 3・問 2

実際に **AdaBoost** のプログラムを作成し，実験データ（例えば mushroom）を用いて，その性能などを実験し，その結果・解析・考察を述べよ．訓練データ（つまり事例集合）として 1000 個くらいを使い，残りのデータを使って得られた仮説の良さを評価してみるとよい．ベストな仮説は何か？訓練データを多くするとどうなるか？高速化の工夫と効果は？等々，いろいろと調べられると思う．

c 言語でアルゴリズムの実装は python より難しいため，今回は **pyclassic** のソースを参照し，python で **Adaboost** のプログラムを作成した，参考先は `http://code.google.com/p/pyclassic/`．アルゴリズムとしては，資料通りの伝統的な Adaboost を使う，たくさんの識別器も選ばれるが，ここでは決定株（Decision stump）という弱識別器だけを考える．

また，実験データは mushroom を用いる，今回のデータは shuffle 済みなので，訓練集合をデータの前からの m 個，テスト集合をデータの後からの n 個，$m + n <$ データの数 とする．ここでは，$m \in \{500, 1000, 1500, 2000\}$，$n \in \{500, 1000, 1500, 2000\}$，それぞれの m, n を組み合せ，訓練・予測を行う．閾値 $\varepsilon = 0.1$，反復回数 $T = 10$ に設定する，実験結果は下記である．

表 1　データサイズ別の判別精度

訓練集合サイズ	テスト集合サイズ	判別精度 (%)	学習時間 (s)	予測時間 (s)	反復回数
500	500	0.822000	00.224870	00.000334	3
500	1000	0.837000	00.191105	00.000409	3
500	1500	0.838667	00.225717	00.000910	3
500	2000	0.842000	00.195486	00.001162	3
1000	500	0.826000	00.451063	00.000326	3
1000	1000	0.841000	00.408634	00.000505	3
1000	1500	0.844667	00.414288	00.000487	3
1000	2000	0.848500	00.375509	00.000639	3
1500	500	0.826000	00.574550	00.000239	3
1500	1000	0.841000	00.537895	00.000383	3
1500	1500	0.844667	00.562652	00.000431	3
1500	2000	0.848500	00.548065	00.000555	3
2000	500	0.826000	00.713411	00.000239	3
2000	1000	0.841000	00.703578	00.000339	3
2000	1500	0.844667	00.705330	00.000442	3
2000	2000	0.848500	00.693032	00.000532	3

訓練集合サイズだけを考察すると，500 個の集合より 1000 個以上のほうが精度が高い，しかし，1000 個，1500 個，2000 個の訓練集合の精度が等しい，オッカムのカミソリより，コンパクトな仮説のほうが望ましい，機械学習における，簡単なモデルのほうが overfitting しにくいので，1000 個の

訓練集合のほうが効率よいと思う．一方，テスト集合サイズが大ければ大きいほど精度が高いに見られる．また，学習時間は訓練集合サイズとは正相関，学習時間と比べ，予測時間はかなり短いこと（早い過ぎで正確に測られない）も分かられる．反復回数を考えると，すべての学習は 3 反復まで終わるので，閾値に近づくのは早いである．閾値をそれぞれ $0.1, 0.2, 0.3, 0.4$ に設定し，実験をやり直したが，全く同じな結果が出るので，ここでは挙げない．

次は **adaboost** の収束を考え，閾値を外し，$T = 10, \varepsilon = 0.1, m = 1000, n = 2000$ の設定で各反復の計算結果を示す．ただし，$e = P_{\boldsymbol{\alpha}, D_0}[f_*(\boldsymbol{\alpha}) \neq f_t(\boldsymbol{\alpha})]$.

表 2　各反復の予測精度

反復（t）	誤判別率（e）	優位度（γ_t）	仮説の重み（α_t）
1	0.500000	0.347000	0.855631
2	0.366000	0.297096	0.684122
3	0.000000	0.378471	0.989016
4	0.366000	0.306668	0.714251
5	0.000000	0.379926	0.995865
6	0.366000	0.307153	0.715808
7	0.000000	0.380001	0.996219
8	0.366000	0.307178	0.715889
9	0.000000	0.380005	0.996237
10	0.366000	0.307179	0.715893

表通り，誤判別率は 0 反復でランダム誤判別率 0.5 から 2 反復ですぐ 0.000000 に減少するが，閾値を設定せずに計算し続くと，誤判別率が上がり，0.366000 に戻ってしまい．優位度も同じく二つの値の間に繰り返す，むだな仮説も増加する．従って閾値の設定が必要だと思う．なお，資料よりパラメータを代入し，Adaboost のブースティング性を計算すると，反復回数 $T \leq 19.12$ だが，実際にブースティングはかなり早いので，上界までは行かない．

高速化するため，次の方法を考えた．上記の結果より，学習時間の削減をメインに考える．

1. **よりよい弱分類器を使う**　今回は Decision Stump を実装したが，他の弱分類器を使ったほうが早いと思う．例えば，Hard margin の SVM ならば，二次計画問題に定着できる（高次元に射影しなければいけないが）；また，Lasso などの ℓ_1 アルゴリズムを用い，疎性が高い解を求め，計算時間もメモリーの減少できる．一方，Adaboost で各反復で予測を行う時，重みしか更新されないので，もしオンライン学習のアルゴリズムを使い，更新された分だけをアップデートすれば，メモリーが削減できるし，早いスピードで収束することも可能になる．
2. **変数の計算**　例えば，D_t の更新する際，適当な スデップサイズ を追加すればもっと早く計算できると思う．
3. **各反復でのメモリー削減**　今回のプログラムは毎回予測を行う際すべての α_t を使ったが，実際にはメモリーに記録し，更新分だけを追加すればよいと思う．